Standalone Server User Guide

Author	Version	Date	Comments
George Inman	1	19/10/09	First Version
George Inman	1.1	07/12/09	Updated Version
George Inman	1.2	08/12/09	Added endorsed directory Instructions

Table of Contents

Standalone Server User Guide 1
- Introduction 1
- Overview of the Service 2
- Installing the Service 2
- Server Configuration 3
  - The TCPConfiguration element 3
    - General Parameters 3
    - Table 1. Example HTTP Configuration 4
    - SSL Specific Parameters 4
    - Table 2. Example HTTPS Configuration 5
- PDP and CVS Configuration 5
  - Configuring a PERMIS PDP and CVS 6
    - Table 3. PERMIS Policy Configuration Example 6
  - Configuring a Sun PDP 7
    - Table 4. XACML Policy Configuration Example 7
- Configuring a Trust PDP 7
    - Table 5. Trust PDP Configuration example 8
- Testing The Server 8
    - Table 6. Test Service Configuration Example 8
- Protocol Information 8
  - Specifying a CVS policy when making a WS-Trust Request 9
    - Table 7. An Example WS-Trust Request, with referenced Policy Identifier 9
  - Specifying an Authorisation Policy to use when making an XACMLAuthzRequest 9
    - Table 8. An example SAML-XACML Policy reference. 10
    - Table 9. An Example SAML-XACML Request, with referenced Policy Identifier 10
- References 11
- Appendix 1. Server WSDL 11

Introduction

The standalone server is a network accessible application independent Authorisation server that can be used as an application independent PEP or Credential Validation Service to respond to any application's request for an authorisation decision.

The following instructions will teach you how to install and tailor the standalone server to allow you to make authorisation requests across a network using standardised authorisation protocols and receive authorisation responses for use in your application.

Overview of the Service

The standalone server is a Java[1] based application with an embeded Apache Axis2 [2] service that accepts requests for authorisation using three standardised protocols messages sent using SOAP[3]. The first of these protocol languages is XACML [4] which is implemented as a test message handler and should not be used in production environments. The second handler is an XACML over SAML 2.0 [5] message handler, this handler has been produced in accordance with the constrained authorisation profile outlined in [6]. The final handler operates as a Ws-Trust [7] CVS handler which provides the requestor with a SAML assertion containing valid Attributes as specified in [8]. Currently the handlers that supports the use of multiple policies is the XACML over SAML 2.0 message handler and the WS-Trust message handler.

Request messages should be sent to the server's endpoint which will determine the type of the messages based on the XML namespace of the request message. Please note that only messages that conform to the relevant message schemas will be accepted by the service.

Installing the Service

Prior to installation the standalone server has the following requirements:

- A Sun Java Runtime Environment – this should be a 1.6 release of the JRE olderversions are not supported.
- (Optional) If you wish to make the server available over a network then a single port number should be reserved for the service and this port should then be opened in your firewall.
- (Optional) If you wish to run the server using SSL then you may wish to install OpenSSL or similar for use when creating server certificates.

In order to install the service you should download the latest release of the service from the PERMIS website (http://sec.cs.kent.ac.uk/permis) and unzip the release package to a folder of your choice. Once this folder has been unzipped you should open a new terminal window and navigate into the newly unzipped directory. Before the server can be run you must endorse the XML parsers contained in the endorsed directory of the relese. This can be accomplished by copying the endorsed directory in the release to the "lib/" directory of your Java.

For Windows users:

```
C:\...\standalone>: copy endorsed %JAVA_HOME%\lib\
```

For Linux users:

```
...:~/standalone$ cp –R ./endorsed $JAVA_HOME/lib/
```

You should now be ready to test the service by running one of the two following commands:

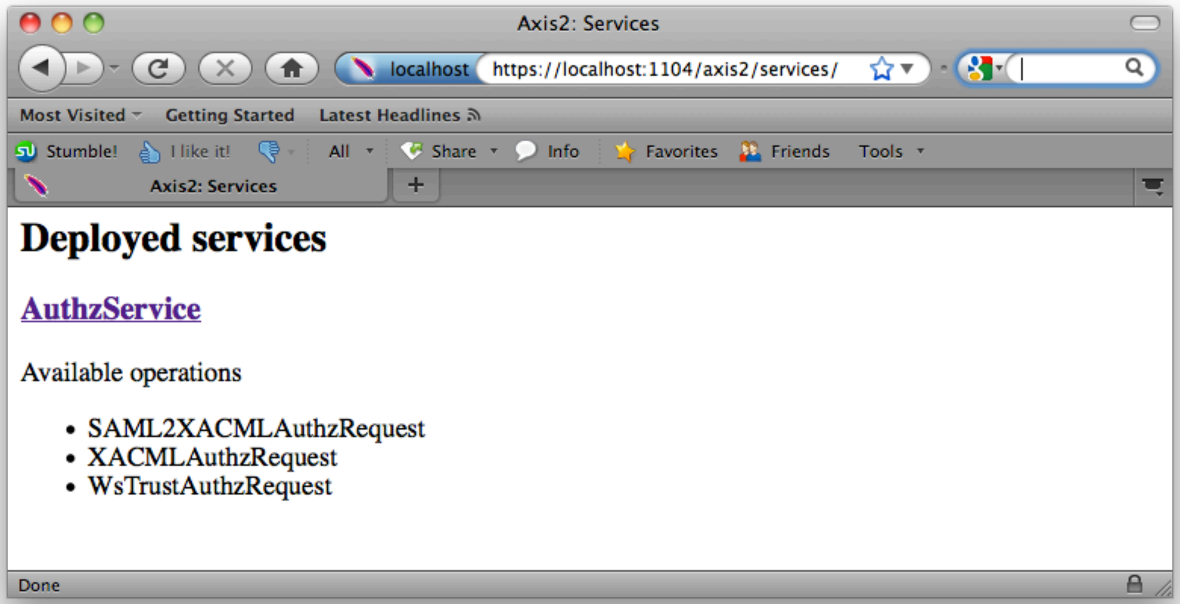
For Windows users:

```
C:\...\standalone>: standalone.bat
```

For Linux users:

```
...:~/standalone$ ./standalone.sh
```

At this point you should be able to verify that the service has been installed properly by navigating to https://localhost:1104/ which should show a page similar to the one displayed below:



Deployed services

AuthzService

Available operations

- SAML2XACMLAuthzRequest
- XACMLAuthzRequest
- WsTrustAuthzRequest

Please ensure that a service named AuthzService has been deployed and that it has three separate operations SAML2XACMLAuthzRequest, XACMLAuthzRequest and WsTrustAuthzRequest.

Please note: Occasionally when started additional operations are made available. If this occurs please restart the server as there is a bug within the Axis WSDL parsing code meaning that the schema is incorrectly loaded occasionally and we are currently working to fix this bug.

At this stage you should have a fully operational standalone PERMIS Authorisation server deployed with two default policies, one of which can be queried using the example SOAP request messages included in the "./Example Request Messages" folder of the release package using some form of Soap client such as SoapUI[9].

Server Configuration

All server and policy configurations are defined in a single file in the root directory of the release package called "permis.xml". This file is consists of a single <PERMISStandaloneConfiguration> element containing a single <TCPConfiguration> element that is used to configure the axis server itself and multiple elements used to configure each indivual message handler type: <PERMISConfiguration> elements that are used to configure individual instances of PERMIS. <SunPDPConfiguration> elements that are used to configure individual instances of the XACML PDP. <TrustPDPConfiguration> elements that are used to configure individual instances of the TrustPDP and <TestService> elements that are used to configure GRANT all or DENY all handlers.

The TCPConfiguration element

At its most basic the TCPConfiguration element defines the port number upon which the server listens, the number of threads to use for requests and the protocol to use. Where required additional configuration parameters are included in order to configure the protocol listener e.g. for SSL. We specify below the possible parameters for this service and their expected contents.

General Parameters

These parameters are required by all server configurations.

<ServerPort> - This element is used to specify the port number on which the server should accept

incoming requests. It takes a single numeric value. If a port has been opened in your firewall to support this service then this value should match the value of that port.

<ThreadCount> - The ThreadCount element is used to specify how may requests can be handled in parallel by the server. The value placed here should be numeric and should vary according to the resources allocated to the system.

<Protocol> - The Protocol element specifies the underlying protocol upon which SOAP requests will be received by the server. This may currently contain a value of either "http" or "https". If the system cannot determine the type of the protocol then it will default to "http".

Table 1. Example HTTP Configuration

```
<TCPConfiguration>
   <ServerPort>1104</ServerPort>
   <ThreadCount>10</ThreadCount>
   <Protocol>http</Protocol>
</TCPConfiguration>
```

Using the above configuration you should be able to navigate to http://localhost:1104 and view the service as before.

SSL Specific Parameters

These parameters are only required when operating using the HTTPS server.

<PrivateKey>- This element should contain a relative or absolute path that can be used to determine a file containing a private key certificate. This certificate will be used to secure the SSL server and may or may not be encrypted. If the file is encrypted, however, the PrivateKeyPassword element must be present.

<PublicKey> - This element should contain a relative or absolute path that can be used to determine a file containing a public key certificate(PKC). The contents of this file must contain a PKC certificate that matches the details contained in the private key file defined above.

<PrivateKeyPassword> - This element should contain a string representation of the password required to access the Private Key file. This element is only required when the private key is protected by a password.

<KeyStore> - This element should contain a relative or absolute path that can be used to create a Java KeyStore file containing the private and public key certificates loaded from their respective files. Please note that this file should not exist prior to service initialisation. **If the file is found to exist it will be overwritten.**

<TrustStore> - This element should contain a relative or absolute path that can be used to locate a Java TrustStore file containing the certificates of server entities that the service trusts. For more information please see Section 4.

<KeyStorePassword> - This optional element should contain a string representation of the password that will be used to access the KeyStore file. If this element is not present the KeyStore password will be assumed to be "password".

<TrustStorePassword> - This optional element should contain a string representation of the password required to access the TrustStore file. If this element is not present the KeyStore password will be assumed to be "password".

<RequireClientAuthentication> - This element specifies whether or not the server should only accept requests from servers with whom it has a pre-existing trust relationship i.e. their SSL certificate is in the TrustStore. A value of “true” specifies that only trusted servers can access the service a value of “false” specifies that any server may make authorisation requests.

Table 2. Example HTTPS Configuration

```
<TCPConfiguration>
   <ServerPort>1104</ServerPort>
   <ThreadCount>10</ThreadCount>
   <Protocol>https</Protocol>

    <!-- SSL only Configurations -->
   <PublicKey>./server.crt</PublicKey>
   <PrivateKey>./server.key</PrivateKey>
   <PrivateKeyPassword>password</PrivateKeyPassword>
   <KeyStore>./keystore.jks</KeyStore>
   <KeyStorePassword>password</KeyStorePassword>
   <RequireClientAuthentication>true</RequireClientAuthentication>
   <TrustStore>./truststore.jks</TrustStore>
   <TrustStorePassword>password</TrustStorePassword>
</TCPConfiguration>
```

If you now navigate to https://localhost:1104/ you should be asked by your browser to provide a certificate for authentication. The example trust store included in the release should contain a single PKC that corresponds to a PKCS#12 file in the release named trusted.p12, which has a password of “password”. You should now be able to see the service as before.

Please Note : The SSL certificates provided with the release should not be used to provide SSL support in deployed systems.

PDP and CVS Configuration

The authorisation server offers support for several PDP implementations including the PERMIS PdP/CVS [9], Sun’s XACML PDP [10] and Eindhoven’s Trust PDP. Each of these configuration types utilise a different type of configuration element in the main configuration file:

- <PERMISConfiguration> elements configure an instance of the PERMIS PDP/CVS.
- <SunPDPConfiguration> elements configure an instance of the Sun XACML PDP.
- <TrustPDPConfiguration> elements configure an instance of the Trust PDP.

If a error occurs whilst configuring these elements the element will be skipped and appropriate error information will be oputputted to the log file.

Please Note: At construction the server will attempt to determine a default PDP/CVS for incoming requests. This default policy will be defined using the isDefault attribute present on all the PDP configuration elements. Only one default policy may be specified in any configuration file instance.

Configuring a PERMIS PDP and CVS

Each <PERMISConfiguration> element defined in the configuration file describes a seperate instance of a PERMIS RBAC server that is accessible through the server's authorisation endpoint. Whilst multiple policies can be configured via this configuration file only one can be used to access the XACML only endpoint or be used as for SAML2XACML and WS-TRUST requests that do not contain a policy identifier. We call this policy the default policy and specify it as a boolean attribute of the PERMISConfiguration element itself called “isDefault”.

Please Note: Whilst the server architecture currently supports the use of Obligations in Policies a user configurable mechanism for enforcing obligations has not yet been implemented. Therefore obligations will only be enforced if the appropriate handling code has been compiled into the server architecture prior to usage.

The possible configuration elements defined for this configuration type are:

<PolicyLocation> - The PolicyLocation element specifies the location of the policy to be used with this service. This may take the form of the URL of an LDAP server, a WebDAV server URL, the path to an Attribute Certificate or the Path to an XML file.

<PolicyIssuer> - the Policy Issuer specifies the Policy writer. When accessing policies which are stored in remote repositories this value is also used to determine the user entry in which the policy is stored.

<PolicyIdentifier> - The Policy Identifier specifies a Unique identifier that can be used to identify the correct policy to be used. This value must match the OID attribute contained in the policy file itself. This value is then used to determine which policy to load from repositories and later this identifier can be used when making both Ws-Trust and SAML-XACML requests to determine which policy to use for credential validation or authorisation.

<LDAPACAttribute> - This element specifies the LDAP attribute name that is used to hold Attribute Certificates for authorisation.

<LDAPPKCAttribute> - This element specifies the LDAP attribute name that is used to hold user PKCs for signature verification.

<CredentialLocation> - The Credential Location element is used in pull mode to define the repositories from which user credentials should be pulled. This element should take the value of an LDAP or WebDAV attribute repository's URL.

<RootPKC> - This element specifies the paths to certificate authoritys that can be used when verifying user certificates and signed credentials.

Table 3. PERMIS Policy Configuration Example

```
<!-- example Policy Configuration using XML and no Signature Verification.
   This policy is the default policy as specified by the isDefault attribute-->
        <PERMISConfiguration isDefault="true">
                <!-- The location of the policy -->
     <PolicyLocation>./policy.xml</PolicyLocation>
                <!-- The issuer of the policy -->
     <PolicyIssuer>cn=A Permis Test User,o=Permisv5, c=gb</PolicyIssuer>
```

```
                <!-- For XML policies the Policy Identifier may have any unique value but MUST
still be set -->
                <PolicyIdentifier>TestPolicy</PolicyIdentifier>
                <!-- The LDAP attribute where the users policy/attributes are stored -->
      <LDAPACAttribute>attributeCertificateAttribute</LDAPACAttribute>
                <!-- The LDAP attribute where the user's PK certificate is stored -->
      <LDAPPKCAttribute>userCertificateAttribute</LDAPPKCAttribute>
                  <!-- The location of user credentials -->
      <CredentialLocation>ldap://sec.cs.kent.ac.uk/c=gb</CredentialLocation>
        </PERMISConfiguration>
```

For additional policy configurations please refer to the example permis.xml configuration file included in the release package.

Configuring a Sun PDP

Each <SunPDPConfiguration> element defined in the configuration file describes a separate instance of a Sun XACML PDP, accessible through the server's authorisation endpoint. We do not currently support SAML XACML or Ws-Trust requests for this PDP type and whilst multiple policies can be configured via this configuration file only one can be used to access the XACML only endpoint. We call this policy the default policy and specify it as a boolean attribute of the SunPDPConfiguration element itself called "isDefault".

Please Note: In order to provide access to a Sun XACML PDP instance it MUST be configured as the default policy.

The possible configuration elements defined for this configuration type are:

<PolicyLocation> - The PolicyLocation element specifies the absolute or relative path location of the policy to be used with this service. This may only take the form of a Path to an XML file containing an XACML 2.0 Policy construct. This element must be present at least once, and may be used multiple times to specify multiple policies.

Table 4. XACML Policy Configuration Example

```
<!-- an example XACML Policy Configuration -->
<SunPDPConfiguration isDefault="false">
        <!-- The location of the Required XACML Policy files -->
        <PolicyLocation>xacmlpolicy.xml</PolicyLocation>
        <PolicyLocation>xacmlpolicy-second.xml</PolicyLocation>
</SunPDPConfiguration>
```

Configuring a Trust PDP

Each <TrustPDPConfiguration> element defined in the configuration file describes a separate instance of a Trust PDP, accessible through the server's authorisation endpoint. We do not currently support SAML XACML or Ws-Trust requests for this PDP type and whilst multiple policies can be configured via this configuration file only one can be used to access the XACML only endpoint. We call this policy the default policy and specify it as a boolean attribute of the TrustPDPConfiguration element itself called "isDefault".

Please Note: In order to provide access to a Trust PDP instance it MUST be configured as the default policy.

The possible configuration elements defined for this configuration type are:

<PolicyConfigFile> - The PolicyConfigFile element is used to provide a relative or absolute path to a TrustPDP policy configuration file that defines the policies required to initalise the PDP instance. The element may only be defined once per TrustPDPConfiguration instance.

<TrustServiceConfigFile> - The TrustServiceConfigFile is used to provide a relative or absolute path to a Trust Service confiuration file which specifies the class names of the required trust services.

Table 5. Trust PDP Configuration example

```
<TrustPDPConfiguration isDefault="true">
   <!-- The location of the Policy Configuration file -->
   <PolicyConfigFile>./config.xml</PolicyConfigFile>
   <!-- The location of the Trust Service Configuration file -->
   <TrustServiceConfigFile>./config-trustservice.xml</TrustServiceConfigFile>
</TrustPDPConfiguration>
```

For more information on configuring a Trust PDP and the contents of the referrenced configuration files please refer to the trust PDP's installation documents.

Testing The Server

The standalone server also provides a testing mechanism that provides both XACML and SAML XACML grant and deny handlers. This service means that as long as the server receives correctly formatted requests then either grant or deny replies will always be received from the default policy endpoints no matter the contents of the request.

The test handlers can be initialised by adding a <TestService> element to the permis.xml configuration file. This element chould have a single attribute "handler" which is used to specify whether the service returns GRANT or DENY responses. For GRANT responses the attribute should have the value "permit" and for deny responses "deny".

Table 6. Test Service Configuration Example

```
<!-- An example GRANT test handler -->
<TestService  handler="grant"/>
```

Please note: This handler overrides any other default policies configured in the permis.xml configuration file and should be omitted in production services.

Protocol Information

Due the the proliferation of different standards and versions of standards we wish to make clear that contrary to previous releases of this software we now only support three distinct message types:

1. xacml-context:Request messages as defined in [4].
2. xacml-samlp:XACMLAuthzDecisionQuery messages as defined in [5].
3. wst:RequestSecurityToken messages as defined in [7] and constrained by [8].

In addition to the standard message types defined in these file we have also implemented standards compliant but otherwise un-profiled means of specifying the CVS or authorisation policies to use when making RequestSecurityToken and XACMLAuthzDecisionQuery messages, we are currently in the process of profiling these requests and standardising them and will make the full profiles available in the near future.

Specifying a CVS policy when making a WS-Trust Request

In order to specify the CVS policy to use when making the WS-Trust Request a request should be specified according to the profile described in [8]. Once this request has been constructed a <wsp:PolicyReference> element should be added to the body of the request. The URI attribute of this element should contain a policy OID that matches a policy OID configured into the main configuration file of the server (permis.xml).

For example the request defined below would mean that a policy with the OID of mysite-policy would be used to provide the credentials for this request.

Table 7. An Example WS-Trust Request, with referenced Policy Identifier

```
<wst:RequestSecurityToken xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512/"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" >
   <wst:TokenType>
      urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML
   </wst:TokenType>
   <wst:RequestType>
      http://schemas.xmlsoap.org/ws/2005/02/trust/validate
   </wst:RequestType>
   <wsp:PolicyReference uri="mysite-policy" />
   <wst:Claims Dialect="http://www.ogf.org/authz/2008/06/CVS/pull">
      <saml:Assertion ID="Permis-Credential-Validation-Service-V1.0" IssueInstant="Wed Oct 14
16:10:15 BST 2009" Version="2.0" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
         <saml:Issuer>
            cn=A Permis Test User,o=PERMISv5,c=gb
         </saml:Issuer>
         <saml:Subject>
            <saml:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:X509SubjectName">
               CN=User0,o=PERMISv5,c=gb
            </saml:NameID>
         </saml:Subject>
       </saml:Assertion>
   </wst:Claims>
</wst:RequestSecurityToken>
```

Specifying an Authorisation Policy to use when making an XACMLAuthzRequest

In order to specify the policy to use when making an XACMLAuthzRequest a request should be specified according to the profile described in [5]. Once this request has been constructed a <Policy> element of type "urn:oasis:names:tc:xacml:2.0:policy:schema:os" should be added to the body of the request. The PolicyId attribute of this element should contain a policy OID that matches a policy OID configured into the main configuration file of the server (permis.xml). The RuleCombiningAlgId attribute of this message should be set to "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides" and an empty target element should be included e.g.

Table 8. An example SAML-XACML Policy reference.

```
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
   PolicyId="mysite-policy"
   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-
overrides">
   <Target/>
</Policy>
```

For example the request defined below would mean that a policy with the OID of mysite-policy would be used to provide the authorisation decision for this request.

Table 9. An Example SAML-XACML Request, with referenced Policy Identifier

```
<XACMLAuthzDecisionQuery
xmlns="urn:oasis:names:tc:xacml:2.0:profile:saml2.0:v2:schema:protocol:cd-01"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:profile:saml2.0:v2:schema:protocol:cd-01
file:/home/sfl/work/issrg/oasis-documents/xacml3/XACML-3.0-cd-1-updated-2009-May-
07/XSD/xacml-2.0-profile-saml2.0-v2-schema-protocol-cd-1.xsd"
   ID="A2009-10-13T12.57.07"
   Version="2.0"
   IssueInstant="2009-10-13T12:58:12.209Z">
<xacml-context:Request xmlns:xacml-
context="urn:oasis:names:tc:xacml:2.0:context:schema:os">
   <Subject xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os">
      <Attribute AttributeId="urn:oid:1.2.826.0.1.3344810.1.1.14"
DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>
            member
         </AttributeValue>
      </Attribute>
   </Subject>
   <Resource xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os">
      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>
            http://www.mysite.com/members/
         </AttributeValue>
      </Attribute>
   </Resource>
   <Action xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os">
      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>GET</AttributeValue>
      </Attribute>
   </Action>
   <Environment xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"/>
</xacml-context:Request>
```

```
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
   PolicyId="mysite-policy"
   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-
overrides">
   <Target/>
</Policy>
</XACMLAuthzDecisionQuery>
```

References

[1] Gosling, J., Joy, B., Steele, G., and Bracha, G. 2005 Java(Tm) Language Specification, the (3rd Edition) (Java (Addison-Wesley)). Addison-Wesley Professional.
[2] Apache Axis 2 project, see http://ws.apache.org/axis2/
[3] Gudgin, M., Hadley, M., Mendelsohn, M. et al. (2003), 'SOAP Version 1.2 Part 1: Messaging Framework', W3C Recommendation, 24th June (URL: http:// www.w3.org/TR/2003/REC-soap12-part1 – 20030624/).
[4] OASIS, "OASIS eXtensible Access Control Markup Language (XACML) Version 2.0", OASIS Standard, 1 February 2005.
[5] OASIS, "SAML 2.0 Profile of XACML, Version 2", Committee Draft 1, 16 April 2009
[6] David W Chadwick, Linying Su, Romain Laborde , "Use of XACML Request Context to Obtain an Authorisation Decision", OGSA Standard, September 2006.
[7] S. Anderson et al., "Web Services Trust Language (WS-Trust)," technical report, 2005.
[8] David Chadwick, Linying Su, Use of WS-TRUST and SAML to access a Credential Validation Service, OGSA Draft, June 2009.
[9] PERMIS PDP/CVS, see http://sec.cs.kent.ac.uk/permis or http://www.openpermis.org/
[10] Sun's XACML PDP, see http://sunxacml.sourceforge.net/

Appendix 1. Server WSDL

This appendix contains a copy of the wsdl used to generate the Permis Standalone server's message handling code. For additional schema information please refer the the "resources" folder in the release package which contains a copy of this wsdl as well as all the schema used to generate the service.

```
<wsdl:definitions targetNamespace="http://sec.cs.kent.ac.uk/authzservice"
xsi:schemaLocation="http://schemas.xmlsoap.org/wsdl/
http://schemas.xmlsoap.org/wsdl/wsdl.xsd      http://www.w3.org/2001/XMLSchema
http://www.w3.org/2001/XMLSchema.xsd" xmlns:xacml-
policy="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xacml-
saml="urn:oasis:names:tc:xacml:2.0:profile:saml2.0:v2:schema:assertion:cd-01" xmlns:xacml-
context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:ws="http://www.example.com/webservice" xmlns:wst="http://docs.oasis-open.org/ws-
sx/ws-trust/200512/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:tns="http://sec.cs.kent.ac.uk/authzservice"
xmlns:wsoap="http://www.w3.org/2004/08/wsdl/soap12"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xacml-
samlp="urn:oasis:names:tc:xacml:2.0:profile:saml2.0:v2:schema:protocol:cd-01"
```

```
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <wsdl:types>
    <xsd:schema targetNamespace="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:import namespace="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
schemaLocation="access_control-xacml-2.0-policy-schema-os.xsd"/>
        </xsd:schema>
    <xsd:schema targetNamespace="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:import namespace="urn:oasis:names:tc:xacml:2.0:context:schema:os"
schemaLocation="access_control-xacml-2.0-context-schema-os.xsd"/>
        </xsd:schema>
    <xsd:schema targetNamespace="http://docs.oasis-open.org/ws-sx/ws-trust/200512/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:import namespace="http://docs.oasis-open.org/ws-sx/ws-trust/200512/"
schemaLocation="http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3.xsd"/>
        </xsd:schema>
    <xsd:schema
targetNamespace="urn:oasis:names:tc:xacml:2.0:profile:saml2.0:v2:schema:assertion:cd-01"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:import
namespace="urn:oasis:names:tc:xacml:2.0:profile:saml2.0:v2:schema:assertion:cd-01"
schemaLocation="xacml-2.0-profile-saml2.0-v2-schema-assertion-cd-1.xsd"/>
        </xsd:schema>
    <xsd:schema
targetNamespace="urn:oasis:names:tc:xacml:2.0:profile:saml2.0:v2:schema:protocol:cd-01"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:import
namespace="urn:oasis:names:tc:xacml:2.0:profile:saml2.0:v2:schema:protocol:cd-01"
schemaLocation="xacml-2.0-profile-saml2.0-v2-schema-protocol-cd-1.xsd"/>
        </xsd:schema>
    <xsd:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:import namespace="urn:oasis:names:tc:SAML:2.0:protocol" schemaLocation="saml-
schema-protocol-2.0.xsd"/>
        </xsd:schema>
    <xsd:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:import namespace="urn:oasis:names:tc:SAML:2.0:assertion" schemaLocation="saml-
schema-assertion-2.0.xsd"/>
        </xsd:schema>
  </wsdl:types>
  <wsdl:message name="saml2XACMLAuthzRequestMessage">
    <wsdl:part name="parameters" element="xacml-samlp:XACMLAuthzDecisionQuery">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="WsTrustAuthzResponseMessage">
    <wsdl:part name="parameters" element="wst:RequestSecurityTokenResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="WsTrustAuthzRequestMessage">
  <wsdl:part name="parameters" element="wst:RequestSecurityToken">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="saml2XACMLAuthzResponseMessage">
  <wsdl:part name="parameters" element="samlp:Response">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="xacmlAuthzRequestMessage">
  <wsdl:part name="parameters" element="xacml-context:Request">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="xacmlAuthzResponseMessage">
  <wsdl:part name="parameters" element="xacml-context:Response">
  </wsdl:part>
</wsdl:message>
<wsdl:portType name="AuthzInterface">
  <wsdl:operation name="XACMLAuthzRequest">
    <wsdl:input message="tns:xacmlAuthzRequestMessage">
  </wsdl:input>
    <wsdl:output message="tns:xacmlAuthzResponseMessage">
  </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="WsTrustAuthzRequest">
    <wsdl:input message="tns:WsTrustAuthzRequestMessage">
  </wsdl:input>
    <wsdl:output message="tns:WsTrustAuthzResponseMessage">
  </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SAML2XACMLAuthzRequest">
    <wsdl:input message="tns:saml2XACMLAuthzRequestMessage">
  </wsdl:input>
    <wsdl:output message="tns:saml2XACMLAuthzResponseMessage">
  </wsdl:output>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="AuthzSoapHttpBinding" type="tns:AuthzInterface">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="XACMLAuthzRequest">
    <soap:operation soapAction="urn:oasis:names:tc:xacml:2.0:policy:schema:os"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="WsTrustAuthzRequest">
    <soap:operation soapAction="http://schemas.xmlsoap.org/ws/2005/02/trust"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
```

```
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SAML2XACMLAuthzRequest">
    <soap:operation
soapAction="urn:oasis:names:tc:xacml:2.0:profile:saml2.0:v2:schema:protocol:cd-01"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
 </wsdl:binding>
 <wsdl:service name="AuthzService">
  <wsdl:port name="AuthzEndpoint" binding="tns:AuthzSoapHttpBinding">
    <soap:address location="https://localhost:1104/axis2/services/AuthzService.AuthzEndpoint/"/>
  </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```